09/521,308

First Hit Fwd Refs

☐ Generate Collection Print

L23: Entry 5 of 6 File: USPT Mar 5, 2002

DOCUMENT-IDENTIFIER: US 6353818 B1
TITLE: Plan-per-tuple optimizing of database queries with user-defined functions

Abstract Text (1):
A method, apparatus, and article of manufacture for executing a database query in a database management system. The method comprises the steps of generating a plurality of query plans for the database query, evaluating the plurality of query plans using a measured value for the resource metric, selecting a query plan from the evaluated query plans based on the measured resource metric, and executing the selected query plan. The apparatus comprises a query plan generator for generating a plurality of query plans, each query plan optimized with respect to at least one resource metric, and a query plan evaluator, communicatively coupled to a resource object and a database management system node, the evaluator selecting a query plan from the optimized query plan according to a measured resource metric obtained from the global resource object. The article of manufacture comprises a program storage device tangibly embodying one or more programs of instructions executable by the computer to perform the method steps of executing a database query in a database management system, the method steps comprising the method steps above.

Brief Summary Text (6):
At the same time, wider varieties of data are available for storage and retrieval. In particular, multimedia applications are being introduced and deployed for a wide range of business and entertainment purposes, including multimedia storage, retrieval, and content analysis. Properly managed, multimedia information technology can be used to solve a wide variety of business problems.

Brief Summary Text (7):
For example, multimedia storage and retrieval capability could be used to store check signature images in a banking system. These images may then be retrieved to verify signatures. In addition, the authenticity of the signatures could be confirmed using content-based analysis of the data to confirm that the customer's signature is genuine. However, practical limitations have stymied development of large multimedia database management systems. Multimedia database information can be managed by ordinary relational database management systems (RDBMS), or by object-oriented database management systems (OODBMS). Each of these options present problems that have thus far stymied development.

Brief Summary Text (8):
While object-oriented database management systems are used in the "complex" object and long-running transaction sector, they are generally impractical for mainstream commercial use because they require a large initial capital investment and are incompatible with existing RDBMSs. Further, maintaining two separate data repositories in a RDBMS and a OODBMS is inconsistent with the database management philosophy of maintaining a secure consistent central repository for all data. RDBMSs such as the TERADATA.RTM. system are vastly more popular than OODBMS. However, existing RDBMSs cannot effectively handle large objects, such as multimedia objects. Also, although RDBMS database features and functions apply equally well to alphanumeric or multimedia data types, multimedia objects introduce new semantics problems, and require new strategies for manipulating and moving extremely large objects, which would otherwise overwhelm RDBMS computational

capacity and the I/O capability of the computer implementing the RDBMS.

Brief Summary Text (9):
Content-based analysis of multimedia data in a database management system is also problematic. Multimedia data objects are usually large. For example, even using compression technologies, a 100 minute audio compact disc may contain as much as 132 Mbytes, and a 100 minute VHS movie may contain as much as 1.125 GBytes of data. Improving technologies will further increase data storage and processing requirements. For example, a 100-minute High Definition Television (HDTV) movie will require about 22.5 GBytes of storage. Aggravating this problem further is the fact that many content based algorithms are computationally intensive.

Brief Summary Text (10):
In general, SQL optimizers require and/or use cost estimation, tuple statistics, column demographics, histograms, indexes and/or sampling to optimize queries. Large multimedia databases where large objects (LOBs) are utilized in predicates via user defined functions (UDFs) introduce new load balancing and skew problems for any one of the managed resources. Estimation of UDFs execution costs is an effective technique when the average execution time has low variance, but estimation is less effective when the execution costs are highly variant.

Brief Summary Text (11):
One method of estimating UDF execution costs is by sampling. This technique involves running a random instance of every UDF in the query and using this information to generate an efficient query plan. Sampling is easy to implement, but assumes that a UDF execution time is uniform across all objects. If sampling is used on a UDF with large execution cost variance, then non-optimal plans may be generated.

Brief Summary Text (12):
Regardless of the technique used to generate a query plan (e.g. sampling or historical data), if the cost variance varies widely per tuple, then an efficient (optimal) plan may not be possible to generate at compile-time. Also, an efficient execution plan depends on how the system resources (processing capacity, memory, disk and network) are being used by all active queries. This affects not only high variant UDFs, but expensive uniform ones too. This variance (or expense) is relative to any one or more managed system resources. For example, a UDF that does content analysis on a video column where video lengths in the column span from 2 minutes in length to over 2 hours (note that this says nothing about the skew pattern within the column itself relative to distribution of sizes of videos).

Brief Summary Text (14):
The disclosed embodiment uses a plan-per-tuple optimization paradigm. This plan-per-tuple methodology is most useful for large objects used as predicate-based terms when a non co-located join is involved in the query, but is also useful for non co-located join operations as well. In plan-per-tuple optimization, the execution engine chooses from among N! resource optimization strategies; where N normally represents system manageable resources. In the illustrated embodiment, the N resources selected include: (i) interconnect saturation levels, (ii) available physical memory, (iii) central processing unit (CPU) utilization, and (iv) available disk spool space percentages, but this technique can be applied to any managed system resources. Using the techniques described in this disclosure, the optimizer search space does not include these N! resource optimization strategies. Instead, a plurality of query plans is generated, and the selected query plan is determined by execution engine run-time optimization strategies. When the optimizer identifies an expensive, or more importantly, a high variant user-defined function in the predicate (via collected statistics or historical information), the optimizer generates plans that incorporate plan-per-tuple optimization for that particular compiled query. By using a run-time execution strategy based on system resource availability, not compile-time optimizer search strategies, the present

invention allows optimal queries to be selected for highly variant functions on large or continuously streamed objects.

Brief Summary Text (18):
The apparatus comprises a query plan generator for generating a plurality of query plans, each query plan optimized with respect to at least one resource metric, and a query plan evaluator, communicatively coupled to a resource object and a database management system node, the evaluator selecting a query plan from the optimized query plan according to a measured resource metric obtained from the global resource object.

Detailed Description Text (18):
The resolver 112 obtains statistical, static cost, and historical usage information from the Global Data Dictionary (GDD) 126, which is stored in the form of GDD tables in the RDBMS 124, and managed by the GDD manager 114. The GDD manager 114 also maintains a user-configurable and definable GDD cache that is updated when changes are made to the GDD 126. This is accomplished by spooling entries during data definition language (DDL) M-SQL operations or checking a time-stamp or version number associated with the GDD tables. If a change has occurred, the GDD manager 114 updates the GDD cache. The GDD cache structure is subdivided into boundary areas that are allocated to a GDD table whose values are replaced using a least recently used (LRU) algorithm. This method decreases system response times and improves overall performance. The GDD manager 114 also checks the integrity of the entries in the GDD 313.

Detailed Description Text (20):
The plan generator 118 translates the optimized parse tree into a query execution plan. This is performed by writing functions that transform parse-tree objects into multimedia step (M-Step) commands that can be understood by the component modules that will receive these commands. For example, the RDBMS M-Step can be SQL and for TERADATA.RTM. internal (AMP) steps and the object server 128 may use special object M-Steps that invoke UDF functions. The protocol for M-Step commands is stored in the M-Steps Definition and accessed by the plan generator module 116. Of course, those skilled in the art will recognize the present invention could be practiced with other command protocols as well.

Detailed Description Text (23):
FIG. 2 is a diagram illustrating the operation of the optimizer 118 and evaluator 120 of the present invention in further detail. As set forth above, there are two serious problems with performing functions on large data objects: expensive functions and variant function. The first problem involves the performance of "expensive" functions. In this context, an "expensive" function is one whose performance demands significant database management system 100 resources. This includes functions (user-defined or otherwise) that include computationally intense operations, regardless of the object operated upon, and functions calling for a large number of repeated operations. "Expensive" functions also include those that, although not ordinarily computationally intense, are demanding of system communication, memory, and processing requirements when applied to large objects.

Detailed Description Text (25):
The optimizer module 118 comprises a resource requirement predictor 202. The resource requirement predictor predicts the resource requirements for the database query, reads statistics/demographics 208, and determines if the database query from the user implicates the invocation of expensive or substantially variant UDFs. The statistics and demographics 208 regarding resource metrics associated with the object and function are typically stored in the GDD 126, but may be stored elsewhere in the database management system (for example, with the functions or objects themselves). The statistics/demographics can also be augmented with information from a historical useage of the UDF or the object 210. The source of the statistical knowledge of resource usage by the UDF can be obtained in a variety

of ways. For example, the optimizer module 118 may determine the memory resource useage statistics of a UDF on a large object column by tracking access heap or buffer spool space while the database management system is processing the LOB. Or, the CPU, disk 136 accesses, and disk spool space required per UDF invocation, which indirectly indicate interconnect cost, may be tracked.

Detailed Description Text (26):
FIG. 2 illustrates a resource state with four member resource metrics: processing requirements, denoted R.sub.CPU, volatile memory requirements, denoted R.sub.MEM, non-volatile memory requirements, denoted R.sub.DISK, and communication requirements, denoted R.sub.NET.

Detailed Description Text (28):
Many different optimization algorithms and optimization criteria are within the scope of the present invention. For example, parse object trees may be optimized with regard to any or all of the following criteria: query response time, system throughput, primary network traffic and usage, temporary result and space management, parallel or concurrent execution of query steps, predicate evaluations involving one or more UDFs, and resource allocation provided to individual queries. Also, although a system in which the optimization precedes plan generation, other implementations are within the scope of the current invention. The process of optimization can be performed any time after binding, and different data structures may indicate that optimization should occur after the query plans are generated.

Detailed Description Text (34):
FIG. 3 is a flow chart showing the operations used to practice one embodiment of the present invention. First, as shown in block 302, the computer 102 accepts a database query. Next, the resource requirements to perform the query are estimated, as shown in block 304. As described herein, this can consider either the expense of the database operation, the variance of the database operation, or both factors. The expense of the operation can include statistics regarding the operation, the object, or both. Next, the estimated resource requirements are compared to a threshold value to determine whether execution-time query optimization techniques are required. If the resource requirements compare favorably to the threshold value, execution time query optimization techniques are not required, and a query plan is generated and executed. These operations are depicted in blocks 304-310.

Detailed Description Text (40):
The "FindTumor(MRI)" function is a UDF that examines an MRI and returns a value related to the probability that a tumor was detected in the MRI object data. To estimate the query resource requirements for this function, resource metrics such as the processing, volatile memory, non-volatile memory, and communication requirements implicated by this query are estimated by the optimizer 118, using statistical/demographic data 208 stored in the GDD 126. The optimizer then decides whether to proceed with a single query plan or multiple query plans based upon a comparison between these estimated resource metrics and a threshold value derived from the baseline capabilities of the DMBS nodes 104. The threshold for each resource metric (or weighted combination thereof) can be selected to assure that the optimizer makes a minimum cost decision whether to resort run time optimization. If necessary, the threshold for this decision can be made adaptive by keeping track of the decision errors, and modifying the threshold to minimize them, and by relating the resource metric with object data values. By way of example, a simple resource predictor technique may comprise a table of threshold values relating the size of the object data and the function to be applied to the object data.

Detailed Description Text (42):
FIG. 4 is a flow chart presenting an illustrative embodiment of the process steps used to select the query plan from the evaluated plurality of query plans. In one embodiment, the evaluator 120 of the present invention compares query plans

optimized for certain resource metrics (or combinations thereof), and selects the query plan based on measured (or predicted) values for the resource metrics obtained from the DBMS nodes 104. However, the present invention can be practiced on database systems with multiple DBMS nodes 104. In such cases, the present invention can perform a further runtime optimization by considering selecting the optimal query/DBMS node 104 pairing, and by routing the query to the selected node for execution. For example, the query plans generated for the FindTumor function discussed above may include a query plan that is optimized to minimize processing requirements, while another may be optimized to minimize memory requirements. A first DBMS node 104, such as object server 128A may report resource metrics indicating that it has little available processing capability, but plenty of memory. A second DBMS node 104, such as object server 128B, may report resource metrics indicating that it has little memory available, but it is short on processing capacity. In this case, the evaluator 120 would select one of the two query plans and route that query plan to the appropriate DBMS node 104. When selecting among query plans and DBMS node 104 pairings, the evaluator 120 may generate a value indicative of the expected "cost" of each of the two options, and select the lowest "cost" option.

Detailed Description Text (47):
The apparatus comprises a query plan generator for generating a plurality of query plans, each query plan optimized with respect to at least one resource metric, and a query plan evaluator, communicatively coupled to a resource object and a database management system node, the evaluator selecting a query plan from the optimized query plan according to a measured resource metric obtained from the global resource object.

CLAIMS:

1. A method of executing a database query in a database management system, comprising the steps of:

(a) generating a plurality of query plans for the database query, each query plan optimized with respect to at least one system resource metric of the database management system;

(b) at run time, evaluating the plurality of query plans using a measured value for the system resource metric;

(c) selecting a query plan from the evaluated plurality of query plans based on the measured value; and

(d) executing the selected query plan in the database management system; predicting at least one resource requirement for the query; generating a plurality of query plans for the database query, each query plan optimized with respect to at least one system resource metric of the database management system; at run time, evaluating the plurality of query plans using a measured value for the system resource metric; selecting a query plan from the evaluated plurality of query plans based on the measured value; executing the selected query plan in the database management system; when the predicted resource requirement exceeds a threshold value, and generating a query plan when the predicted resource requirement does not exceed the threshold value.

8. The method of claim 1, wherein the database management system comprises a plurality of database management system nodes and wherein:

the step of evaluating the plurality of query plans using a measured value for the system resource metric comprises the step of obtaining the measured value of the system resource metric for each of the database management system nodes; and the step executing the query plan comprises the step of executing the selected query plan at a database management system node selected according to the measured value for the system resource metric at the database management system node.

9. An apparatus for executing a database query in a database management system, comprising:

a query plan generator for generating a plurality of query plans, each query optimized with respect to at least one system resource metric of the database management system; and

a query plan evaluator, communicatively coupled to a global resource object and a database management system node, the query plan evaluator for evaluating and selecting a query plan from the plurality of query plans at run time according to a measured system resource metric obtained from the global resource object means for predicting at least one resource requirement for the query; means for determining when the predicted resource requirement exceeds a threshold value; and means for generating a query plan when the predicted resource does not exceed the threshold value.

10. An apparatus for executing a database query in a database management system, comprising:

(a) means for generating a plurality of query plans for the database query, each query plan optimized with respect to at least one system resource metric of the database management system;

(b) means for evaluating the plurality of query plans at run time using a measured value for the system resource metric;

(c) means for selecting a query plan from the evaluated plurality of query plans based on the measured value; and

(d) means for executing the selected query plan in the database management system means for predicting at least one resource requirement for the query; means for determining when the predicted resource requirement exceeds a threshold value; and means for generating a query plan when the predicted resource does not exceed the threshold value.

17. The apparatus of claim 10, wherein the database management system comprises a plurality of database management system nodes and wherein:

the means for evaluating the plurality of query plans using a measured value for the system resource metric comprises a means for obtaining the measured value of the system resource metric for each of the database management system nodes; and

the means for executing the query plan comprises a means for executing the selected query plan at a database management system node selected according to the measured value for the system resource metric at the database management system node.

18. A program storage device, readable by computer having a processor and a memory, tangibly embodying one or more programs of instructions executable by the computer to perform the method steps of executing a database query in a database management system, the method steps comprising the steps of:

(a) generating a plurality of query plans for the database query, each query plan optimized with respect to at least one system resource metric of the database management system;

(b) at run time, evaluating the plurality of query plans using a measured value for the system resource metric;

(c) selecting a query plan from the evaluated plurality of query plans based on the measured value; and

(d) executing the selected query plan in the database management system; predicting at least one resource requirement for the query; generating a plurality of query plans for the database query, each query plan optimized with respect to at least one system resource metric of the database management system; at run time, evaluating the plurality of query plans using a measured value for the system resource metric; selecting a query plan from the evaluated plurality of query plans based on the measured value; executing the selected query plan in the database management system, and when the predicted resource requirement exceeds a threshold value, generating a query plan when the query resource requirement does not exceed the threshold value.

25. The program storage device of claim 18, wherein the database management system comprises a plurality of database management system nodes and wherein:

the method step of evaluating the plurality of query plans using a measured value for the resource metric comprises the method step of obtaining the measured value of the system resource metric for each of the database management system nodes; and

the method step of executing the query plan comprises the method step of executing the selected query plan at a database management system node selected according to the measured value for the system resource metric at the database management system node.